Evolution : Vie Artificielle

Introduction

On souhaite créer une application simulant l'évolution d'un univers simple dans lequel se trouvent des êtres vivants : de l'herbe, des moutons et des loups. L'évolution de l'univers sera simulée par une horloge à temps discret qui compte des mois. La population de cet univers simple évoluera de mois en mois suivant des règles générales simples et des règles particulières à chaque catégorie d'être vivant.

Univers d'évolution

L'univers à la même forme que celui utilisé pour les automates cellulaires 2D : il s'agit d'un plateau carré dont les bords sont connectés 2 à 2 (le haut rejoint le bas, la gauche rejoint la droite). Chaque case peut contenir plusieurs êtres vivants (au max 3), et possède de plus un potentiel en sels minéraux.

Il y a bien sûr un aspect héritage et polymorphisme, car il y a bien une classe abstraite "être vivant" qui propose un certain nombre de méthodes virtuelles pures.

Vie des êtres vivants

Tous les êtres vivants mangent : l'herbe "mange" des sels minéraux, les moutons mangent de l'herbe, les loups mangent des moutons. Un être vivant en mange un autre si deux de ces êtres vivants se trouvent sur la même case après s'être déplacé.

Tous les êtres vivants se reproduisent : l'herbe se reproduit à l'identique ou apparaît sur une case vide si elle a assez de sels minéraux, et ce tous les mois, les moutons et les loups se reproduisent une fois par an (attention donc à l'horloge qui compte de mois en mois !), a condition que deux animaux de même catégorie se trouvent sur la même case à la date de reproduction (on ne fait pas apparaître de loupton ou de mouloup !).

Les moutons et les loups se déplacent tous les mois, soit pour chercher de la nourriture, soit pour chercher un partenaire pour se reproduire, éventuellement les moutons peuvent chercher à fuir un loup.

Rayon de visibilité : les loups et les moutons ont une vision limitée à un certain nombre de cases.

Mort des êtres vivants

L'herbe meurt si elle est mangée par un mouton (mais réapparait dès le mois suivant s'il y a assez de sels minéraux dans la case), le mouton meurt s'il est mangé par un loup ou

s'il ne trouve pas à manger pendant 2 mois, ou de vieillesse au bout de 4 ans; un loup meurt s'il ne trouve pas à manger pendant 2 mois ou de vieillesse au bout de 5 ans.

Lorsqu'un animal meure, il fournit des sels minéraux sur la case où il meurt ainsi que sur les 8 cases adjacentes (cela se fait sans délai). Cas particulier : lorsqu'un mouton se fait manger, il ne donne que la moitié des sels minéraux qu'il donnerait en mourant de faim ou de vieillesse.

Utilisation de Design Patterns

Vous pourrez utiliser le Design Pattern "Stratégie" exposé dans le cours de C++ pour faire en sorte que les êtres vivants aient des stratégies différentes et dynamiques (vous pouvez avoir des moutons "idiots" et des moutons "intelligents", idem pour les loups)

Paramètres libres pour effectuer la simulation

Il vous appartient de choisir les paramètres suivants pour le déroulement de la simulation :

- Taille de l'univers;
- Quantité de sels minéraux initiale dans chaque case;
- Nombre de moutons;
- Nombre de loups;
- Stratégie de déplacement (et nombre de case de déplacement) des animaux;
- Stratégie de reproduction s'il y a des animaux "idiots" et "intelligents" (peut-on voir apparaître une sélection naturelle ?);
- Zone de visibilité des animaux.
- Quantité de sels minéraux restituée lors de la mort d'un animal.

Il est tout à fait possible de discuter avec vos camarades de promotion à propos du jeu de paramètres à utiliser pour avoir une simulation intéressante. A priori, les différents phénomènes devraient s'équilibrer : s'il y a beaucoup d'herbe au départ, il devrait y avoir beaucoup de moutons, et donc les loups vont survivre; il s'ensuit une grande consommation d'herbe, la population de mouton va décroître car ils vont manquer d'herbe et se faire manger par les loups; puis la population de loups risque de décroître car il n'y aura pas assez de moutons, etc…

Extensions possibles

Il est tout à fait possible d'imaginer des extensions à de sujet, ajouter d'autres êtres vivants, d'autres types de terrains (plan d'eau pour faire boire les animaux), gérer les saisons (pousse des plantes, froid), gérer l'âge des animaux, différencier mâles et femelles, etc...Toute idée pertinente sera la bienvenue.

L'interface peut tout à fait être réalisée en mode console, cela ne pénalisera aucunement la note.

Affichage /saisie

Evolution doit proposer des affichages en **couleur**

Toutes les saisies effectuées doivent être **sécurisées**

Vous êtes seuls maîtres des informations affichées / à saisir : pensez à l'ergonomie du programme !

Raccourcis clavier

Vous pouvez ajouter des raccourcis clavier pour faciliter l'utilisation de votre application.

Fichiers

Vous devrez utiliser des fichiers pour pouvoir charger / sauvegarder l'univers de jeu avec tous ses paramètres : cela vous permettra d'avoir des simulations reproductibles et de faire de nombreux tests.

Travail à effectuer :

Vous devrez écrire en C++ **bien écrit** l'application Evolution :

- Utilisation propre des classes du C++, avec des fichiers .h et des fichiers .cpp;
- Utilisation de l'héritage, du polymorphisme et des surcharges d'opérateurs;
- Utilisation d'au moins un Design Pattern parmi ceux présentés en cours : singleton, décorateur, stratégie, pool d'objets.

Vous devrez rendre :

Partie application

Vous déposerez sur **Campus EFREI** :

- tous les codes source en C++;
- éventuellement les fichiers nécessaires à la génération de l'exécutable;
- un exécutable;
- les fichiers de données utilisés par l'application (configuration de l'univers)

Le tout dans une archive dont vous contrôlerez l'intégrité. Toute archive vide ou corrompue vaudra 0.

Si vous utilisez une interface graphique évoluée type SDL ou Qt ou autre (**ce qui n'est pas demandé par défaut**), vous veillerez à fournir tous les fichiers nécessaires à un fonctionnement **autonome** de votre application (le correcteur ne devra avoir aucune installation à faire).

Partie rapport

Le rapport de projet contiendra, en plus des rubriques habituelles :

- Un diagramme des classes complété avec tous les membres et méthodes des classes utilisées, à l'exception des constructeurs, destructeurs et opérateurs surchargés ;
- Un mode d'emploi clair de l'application fourni en annexe.

Echéancier

Les soutenances de projet seront programmées pour la fin du mois de Mai (vous avez donc **2 mois** pour réaliser ce projet…). Ce projet est à effectuer en binôme uniquement. La liste des binômes devra être fournie à M. FLASQUE par l'intermédiaire de vos délégués pour le 10 Mai 2009 DERNIER DELAI.